Learning Registry
in 20 Minutes or Less

Welcome to the Learning Registry project! This document will get you rolling with creating, uploading, downloading, and verifying envelopes in and out of Learning Registry server. This tutorial should work on Windows, Linux, and Mac OS X machines.

Additional technical information about the Learning Registry may be found in the Quick Reference Guide.

Contents

Setup

Before you start your 20-minute clock, you’ll need to make sure you’re set up properly. You need to know just enough Unix to be dangerous because you’ll be entering commands into the shell. That means in order to get up and running, you'll need to have Python 2.6.5+ and GPG 2.0.17+ installed and accessible in your path. Depending upon your development platform, you may have to separately download and install some of the additional modules mentioned below.

Important! You must have GPG and Python in your path.

Installation

distribute and pip

1. Install both distribute and pip with the commands below. This assumes you have curl installed. If not, download get-pip.py by other means.

   There are many ways to install pip and Python, including using virtualenv. If you're going to program in Python you may want to consider doing so. However, the following cURL method is simple and should work anywhere:

   ```
   curl -O http://python-distribute.org/distribute_setup.py
   python distribute_setup.py

   curl -O https://raw.github.com/pypa/pip/master/contrib/get-pip.py
   python get-pip.py
   ```

LRSignature

2. Once you have Python, distribute and pip installed, you need to install LRSignature module for python. LRSignature is a Learning Registry resource document signing, validation, and key management module. Install with this command:

   ```
   pip install LRSignature
   ```

Create Key Pair

3. Next you need to create a public and private key pair in GPG:

   ```
   GPG --gen-key
   ```

   - Choose the kind of key you want. Default is "RSA and RSA."
   - Choose any key size. Default is 2048.
   - Choose an expiration. For testing and many applications, "does not expire" is fine.
   - Provide your name, email, and a comment as you wish to be identified with the key.
   - Provide a passphrase. *This step is very important*. It should be hard to guess and you need to remember it without writing it down somewhere anyone can get a copy. The passphrase is what protects your GPG private key which is used to sign your documents on Learning Registry.

Windows

If you're on Windows, you need to find out where your GPG keyring is stored so you can pass that path info to the LRSignature utility. An easy way to find out where the keyring is stored is with this command.

```
GPG --list-keys
```

You'll see the path on the first line of the results. The path is everything but the file name. If you have a file path of:

```
"c:/Users/username/AppData/Roaming/GPG/pubring.GPG"
```

You should have a GPG keyring path similar to:

```
"c:/Users/username/AppData/Roaming/GPG"
```

4. You need to get the fingerprint for your GPG keyset. To find the fingerprint, run

   ```
   GPG --fingerprint
   ```

5. This part is a little annoying. The finger print is listed as "Key fingerprint =". You need to take the string of chars after the fingerprint and remove all the spaces from it to create a fingerprint that you can actually use. In the end it will look something like:

   ```
   E7E350E792BC9A16D7704408FF7F6E95B2A133FA
   ```

6. Next create a file of your public key (substitute YOUR signature below):

   ```
   GPG --export --armor "E7E350E792BC9A16D7704408FF7F6E95B2A133FA" > public-key.txt
   ```

7. You need to upload your public key to a public server somewhere. You can upload it to an SKS server or you can just host it on your own website somewhere. It's simplest for this example just to host the file on a web server somewhere.

   Let's say you're going to host it at `http://myserver.com/GPG/public-key.txt`. Upload that file to your web server in that location. Now your GPG system is all set up.

   Don't forget your passphrase or lose your GPG keyring file. Back them up securely as appropriate. If you lose either one, you can't sign documents with that identity any longer. If someone steals both, they can sign documents that appear to belong to you.

Create the JSON File

The next step is to obtain a test JSON envelope that you can use as a test to upload into the Learning Registry.

8. Save the JSON text below into a text file and save it as "`test.json`". If you wish, edit the file to add custom data elements. For starters, you could add and edit elements in "`keys`" and "`payload_schema`".

   You can read the full description of the envelope in the *Learning Registry Technical Specification* document.

```
{
    "TOS": {
        "submission_TOS": "http://www.learningregistry.org/tos/cc0/v0-5/"
    },
    "active": true,
    "doc_type": "resource_data",
    "doc_version": "0.23.0",
    "identity": {
        "curator": "",
        "owner": "",
        "submitter": "your name or organization here",
        "signer": "your name or org, if you're signing the document",
        "submitter_type": "agent"
    },
    "keys": [
        "science",
        "Newton",
        "apple",
        "what_ever_you_want"
    ],
    "payload_placement": "inline",
    "payload_schema": [
        "hashtags",
        "describing",
        "resource_locator",
        "format"
    ],
    "resource_data": "Put_anything_like_metadata, xml_or_whatever_here",
    "resource_data_type": "metadata",
 "resource_locator": "URI_of_resource"
}
```

   You're now ready to sign this document with LRSignature.

Signing the Document

LRSignature offers two methods to sign your JSON document. The first method creates the signed document and saves to you a local file which you specify. The second method signs the document and then, rather than saving locally, published the signed document directly to the location you specify.

Sign and Save as Local File

9. To sign a document and save the file locally, run the following command, depending on your platform, from within the folder where your test.json document is saved. In this example, the signed document will be named test.signed.json.

   Note: The commands below have been separated into multiple lines of text for readability purposes.

   Mac/Linux

```
cat test.json | python -m LRSignature.cmd  sign ↵
--key "E7E350E792BC9A16D7704408FF7F6E95B2A133FA" ↵
--key-location "http://myserver.com/GPG/public-key.txt"
--passphrase "your secret passphrase" ↵
> test.signed.json
```

   Windows

   Note that the word "type" below is a windows command that you must include.

```
type test.json | python -m LRSignature.cmd  sign
--key "E7E350E792BC9A16D7704408FF7F6E95B2A133FA"
--key-location "http://myserver.com/GPG/public-key.txt"
--passphrase "your secret passphrase"
--gnupghome "path_to_GPG_keyring_from_above"
> test.signed.json
```

   You should now have a new text file "`test.signed.json`" which will be just like the "`test.json`" file except that it will have your signature block in it.

Sign and Upload Document

10. The test.json document may be directly published to Learning Registry public sandbox servers for testing. You can publish to any Learning Registry compatible node using the same technique.

   To sign and publish a document run the following command, depending on your platform, from within the folder where your test.json document is saved.

   Note: The commands below have been separated into multiple lines of text for readability purposes.

   Mac/Linux

```
cat test.json | python -m LRSignature.cmd  sign ↵
--key "E7E350E792BC9A16D7704408FF7F6E95B2A133FA" ↵
--key-location "http://myserver.com/GPG/public-key.txt" ↵
--passphrase "your secret passphrase" ↵
--publish-url "http://sandbox.learningregistry.org/publish"
```

   Windows

   Note that the word "`type`" below is a windows command that you must included:

```
type test.json | python -m LRSignature.cmd sign ↵
```

```
--key "E7E350E792BC9A16D7704408FF7F6E95B2A133FA" ↲
--key-location "http://myserver.com/GPG/public-key.txt" ↲
--passphrase "your secret passphrase"
--publish-url "http://sandbox.learningregistry.org/publish" ↲
--gnupghome "path_to_GPG_keyring_from_above"
```

In both instances you should see the following results though the `doc_ID` value will differ:

```
[{"document_results": [{"OK": true, "doc_ID": "761e70f774634030914fa45617fc8815"}], "OK":
true}]
```

Download Document

11. If you want to get a copy back right away, you can get the document using cURL, any http library, or a a web browser. Once the nodes replicate with each other, ask any node in the network for it.

    If downloading with a browser be sure to include the query string at the end of the URL. In this case, a Boolean value of true indicates that you wish to reference a document directly by it's doc_ID created in the previous upload example.

    Note: The commands below have been separated into multiple lines of text for readability purposes.

Web Browser

```
http://sandbox.learningregistry.org/harvest/getrecord
?request_ID=the doc_ID returned from the upload
&by_doc_ID=true
```

In the case of our example, this URL would be:

```
http://sandbox.learningregistry.org/harvest/getrecord
?request_ID=761e70f774634030914fa45617fc8815
&by_doc_ID=true
```

cURL

To directly save the document run the following command from within the folder where your test.signed.json document is saved. The downloaded document will be named test.download.json.

```
curl -o test.download.json
"http://sandbox.learningregistry.org/harvest/getrecord
?request_ID=761e70f774634030914fa45617fc8815
&by_doc_ID=true"
```

Verifying Signatures

12. Once you obtain a document back from the Learning Registry, you can verify that the signature provided in that document is valid, and that the content that is in the envelope hasn't been changed since the document was signed.

    Note: The commands below have been separated into multiple lines of text for readability purposes.

Mac/Linux

```
cat test.download.json | python -m LRSignature.cmd verify
```

Windows

Note that the word "type" below is a windows command that you must included:

```
type test.download.json | python -m LRSignature.cmd verify
--gnupghome "path_to_GPG_keyring_from_above"
```

In both instances you should see results like this:

```
{"results": [{"resource_locator": "http://resource_locator_url_will_appear_here", "verified":
true}]}
```

OK! That's a full round-trip. You created and uploaded a valid Learning Registry document, and then downloaded a copy back to your local machine.

Slicing Data (Bonus Section)

Let's take a quick look at slicing data. For a more detailed look at slicing, see the *Learning Registry - Slicing* documentation. The "slice" function is an optional service in the Learning Registry that lets you easily pull down a set of documents from a node, based on certain criteria. At the time of this writing you can slice data using the following parameters:

- `identity`: matches documents with value found in any identity sub-field (`submitter, curator, owner, signer`)
- `any_tags`: matches documents with value found in the `keys` field
- `from`: matches documents submitted on date `from` (format: *YYYY-MM-DD*, 1-day granularity)
- `from, until`: matches documents submitted between `from` value (inclusive) and `until` value (non-inclusive).

13. To invoke slice, you basically construct an HTTP GET in the format below. In the Upload Document section the "`test.signed.json`" document with the word "`science`" in the "`key`" array was published. Running the following command you should find at least that one document:

Web Browser

```
http://sandbox.learningregistry.org/slice?any_tags=science
```

cURL

```
curl -o test.science.json "http://sandbox.learningregistry.org/slice?
any_tags=science&ids_only=true"
```

You should get results resembling those below. These results have been separated into multiple lines of text for readability purposes.

```
{
    "replyStart":"2011-09-15 21:10:37.522239",
    "keyCount":1,
    "documents":[{"doc_ID": "761e70f774634030914fa45617fc8815"}],
    "resultCount":1,
        "replyEnd":"2011-09-15 21:10:37.908154"
    }
```

In the case of multiple documents being found the results will resemble:

```
    {
        "replyStart":"2011-09-15 20:24:14.207687",
        "keyCount":1,
        "documents":[
           {"doc_ID": "761e70f774634030914fa45617fc8815", ...etc.},
           {"doc_ID": "85924442d9ab431b93732440940a3636", ...etc.},
           {"doc_ID": "62c2ee17dbcd4899b373cd4cf63ae669", ...etc.}
          ],
        "resultCount":3,
        "replyEnd":"2011-09-15 20:24:15.170966"
    }
```

When `ids_only` is set to `true`, Slice returns a JSON document which includes an array of document ID's matching your parameters. The default value of `ids_only` is false and if not explicitly set to true will return the full documents.

In addition to the `any_tags option you can` submit specific parameters to narrow your search. For example, if you were only interested in documents posted after a specific date you could query based on the date of publication to Learning Registry by using the "`from`" field. The optional "`from`" field has a format of YYYY-MM-DD. For example, "`from=2011-09-15`".

Another optional field is "`identity`". This is the name of the person or organization that is the submitter, author, owner, or curator. An example is "`identity=US Dept of Education`".

If I wished to locate items submitted by "`US Dept of Education" on or after September 15, 2011 it would include "from=2011-09-15`" as well as "`identity=US Dept of Education`". If you combine these options they are ANDed together.

Specifying a value for "`from`" and "`any_tags`" returns documents published on or after "`start_date`" and matching the tags you specify in the "any_tags" field.

Some Learning Registry nodes will have Flow Control enabled for Slice. This means that the node administrator has specified a limit on the number of results returned for a given query. In such cases, when results are returned, they will include a “resumption_token” field. The value of this field is a token that can be used as an argument to reSlice the node, after which the next page of results is returned.

Change Log

Version	Date	Description
1.00	20110616	Initial version
1.01	20110623	Minor cleanup, wording improvements
1.02	20110623	Fixed bug (with temp workaround) in Unix/Mac command lines
1.03	20110916	Added cURL examples. Updated sample code to latest implementation. Added Flow Control information to Slicing. General edits and formatting.
1.04	20110923	Seperated Signing section into local and publish. Correct publish sample code error.

1.05	20110930	Corrected Heading settings.
1.06	20111020	Corrected request_ID case.
1.07	20120201	Replaced references to lrtest02 with references to sandbox.